

A Randomized Algorithm for 3-SAT

Subhas Kumar Ghosh, Janardan Misra*

December 12, 2009

Abstract

In this work we propose and analyze a simple randomized algorithm to find a satisfiable assignment for a Boolean formula in conjunctive normal form (CNF) having at most 3 literals in every clause. Given a k -CNF formula ϕ on n variables, and $\alpha \in \{0, 1\}^n$ that satisfies ϕ , a clause of ϕ is critical if exactly one literal of that clause is satisfied under assignment α . Paturi et. al. (Chicago Journal of Theoretical Computer Science 1999) proposed a simple randomized algorithm (PPZ) for k -SAT for which success probability increases with the number of critical clauses (with respect to a fixed satisfiable solution of the input formula). Here, we first describe another simple randomized algorithm DEL which performs better if the number of critical clauses are less (with respect to a fixed satisfiable solution of the input formula). Subsequently, we combine these two simple algorithms such that the success probability of the combined algorithm is maximum of the success probabilities of PPZ and DEL on every input instance. We show that when the average number of clauses per variable that appear as unique true literal in one or more critical clauses in ϕ is between 1 and $2/(3 \cdot \log(3/2))$, combined algorithm performs better than the PPZ algorithm.

1 Introduction

The problem of finding a satisfiable assignment (SAT) for a propositional formula in conjunctive normal form (CNF) is notably the most important problem in the theory of computation. The decision problem for CNF-SAT was one of the first problems shown to be NP-complete[1, 2]. CNF-SAT is widely believed to require deterministic algorithm of exponential time complexity. A syntactically restricted version of general CNF-SAT is k -SAT, where each clause of a given CNF formula contains at most k literals, for some constant k . k -SAT remains NP complete for $k \geq 3$ (while 2-SAT is solvable in polynomial time [3]). This restriction on the number of literals per clause seem to be of help, and existing algorithms have $\mathcal{O}(2^{\epsilon_k n})$ time complexity for some constant $0 < \epsilon_k < 1$ dependent on k . Several work exists on faster algorithms for k -SAT (cf. [4], [5], [6], [7], [8]).

The objectives of working on k -SAT algorithms are several. Primary of them is to obtain algorithms having provable bounds on the running time that is significantly better than trivial search algorithm (which is $\text{poly}(n) 2^n$ for formula having n variables) and works

*Honeywell Technology Solutions Laboratory, 151/1, Doraisanipalya, Bannerghatta Road, Bangalore, India, 560076, Email:subhas.kumar@honeywell.com, janardan.misra@honeywell.com

for larger set of k -CNF. Second objective is to understand instances that are significantly hard or easy while useful (i.e. they appear in practical problems).

In following we mention all bounds by suppressing the polynomial factors. Monien and Speckenmeyer [5] described first such non-trivial algorithm with running time $\mathcal{O}(2^{(1-\epsilon_k)n})$, with $\epsilon_k > 0$ for all k , and in specific it is $\mathcal{O}(1.618^n)$ for $k = 3$. Faster algorithm for 3-CNF satisfiability is due to Kullmann [9], with running time $\mathcal{O}(1.505^n)$ for $k = 3$. Both of these algorithms are deterministic. Paturi et al. [10] proposed a simple *randomized algorithm* for k -SAT. Though it is not faster than other known algorithms for $k = 3$, it has better performance for larger values of k . This algorithm was improved in [11, 8] with a randomized variant of the Davis-Putnam procedure [12] with limited resolution. Schöning's random walk algorithm [13, 6] is better than [8] for $k = 3$, but is worse for $k \geq 4$. Schöning's random walk algorithm [6] has bound of $\mathcal{O}((2 - 2/(k + \epsilon))^n)$ for some $\epsilon > 0$. Further improvements of his algorithm were found by Hofmeister et al. [14] for $k = 3$. Randomized algorithm of [8] has expected running time $\mathcal{O}(1.362^n)$ for $k = 3$.

Better randomized algorithm is due to Iwama and Tamaki [15], having expected running time $\mathcal{O}(1.3238^n)$ for $k = 3$, which is a combination of the Schöning's random walk algorithm [13, 6] and the algorithm of Paturi et al. [11] (this bound improves to $\mathcal{O}(1.32266^n)$ using modified analysis in [8]). Iwama and Tamaki's algorithm [15] has been improved by Rolf [16] recently to best known randomized bound of $\mathcal{O}(1.32216^n)$ for 3-SAT.

Schöning's algorithm was derandomized in [7] to the currently best known bound of $\mathcal{O}(1.481^n)$ for $k = 3$ and to a bound of $\mathcal{O}((2 - 2/((k + 1) + \epsilon))^n)$ for $k > 3$, using limited local search and covering codes. This was improved for $k = 3$ in [17] to a deterministic bound of $\mathcal{O}(1.473^n)$. Randomized algorithm of [11] was derandomized in [18] for Unique- k -SAT (i.e. k -CNF formulas having only one solution) using techniques of limited independence, i.e. by constructing a small bias probability space to choose samples for original algorithm of [11] yielding deterministic running time $\mathcal{O}(1.3071^n)$ for Unique-3-SAT. In this work we present and analyze a randomized algorithm for finding a satisfiable assignment for a Boolean formula in CNF having at most 3 literals in every clause. We consider the k -SAT algorithm of Paturi et al. [10] for $k = 3$ and combine it with another randomized algorithm that we describe here, such that the success probability of the combined algorithm is maximum of the success probabilities of these two algorithms on every input instance.

Before we proceed further let us introduce some notations. A formula ϕ in n -variables is defined over a set $\{x_1, \dots, x_n\}$. *Literals* are variable x or negated variable $\neg x$. *Clauses* are disjunctions of literals, and we assume that a clause do not contain both, a literal and its negation. A Boolean formula $\phi = \bigwedge_{i=1}^m C_i$ is a k -CNF if each clause C_i is a disjunction of at most k literals. Variables are assigned truth values 1 (TRUE) or 0 (FALSE). An assignment to variables $\{x_1, \dots, x_n\}$ is an element $\alpha \in \{0, 1\}^n$. For $S \subseteq \{0, 1\}^n$ and $\alpha \in S$, α is an *isolated* point of S in direction i if flipping i th bit of α produces an element that is not in S . We will call $\alpha \in S$, j -isolated in S if there are exactly $(n - j)$ neighbors of α in S . An n -isolated point in $S \subseteq \{0, 1\}^n$ will be called isolated.

Given a k -CNF formula ϕ on n variables $\{x_1, \dots, x_n\}$, single iteration of Paturi et al.'s randomized algorithm [10] (see Algorithm-1) works by selecting a random permutation of variables $\pi \in S_n$, and then assigning truth values uniformly at random in $\{0, 1\}$ to each variable $x_{\pi(i)}$ for $i = 1, \dots, n$. However, before assigning a random truth value, algorithm checks if there is an unsatisfied unit clause (i.e., a clause having only one literal) corre-

```

Algorithm PPZ( $\phi$ ) Input: 3-CNF  $\phi = \bigwedge_{i=1}^m C_i$  on variables  $\{x_1, \dots, x_n\}$ 
Pick a permutation  $\pi$  of the set  $\{1, \dots, n\}$  uniformly at random.
for  $i = 1, \dots, n$  do
    if there is an unit clause corresponding to the variable  $x_{\pi(i)}$  then
        | Set  $x_{\pi(i)}$  so that corresponding unit clause is satisfied, let  $b$  be the assignment.
    else
        | Set  $x_{\pi(i)}$  to TRUE or FALSE uniformly at random, let  $b$  be the assignment.
    end
     $\phi := \phi[x_{\pi(i)} \leftarrow b], \alpha_i := b.$ 
end
if  $\alpha$  is a satisfying assignment then
    | return  $\alpha$ .
else
    | return “Unsatisfiable”.
end

```

Algorithm 1: One iteration of procedure PPZ(ϕ)

sponding to variable $x_{\pi(i)}$, and if there is one, it forces the value of $x_{\pi(i)}$ such that the corresponding unit clause gets satisfied. We will call this algorithm PPZ. Let $S \subseteq \{0, 1\}^n$ be the set of all satisfying assignments of ϕ .

Crucial observation made in [10] is that if α is an isolated point of S in some direction i , then there exists a clause in which exactly one literal is satisfied under assignment α – and that literal corresponds to the variable x_i (such a clause will be called *critical* for variable x_i under solution α). Given formula ϕ let $\alpha \in S$ be a fixed satisfying assignment in the set of all satisfying assignments of ϕ . Now observe that after selecting a random permutation of variables π , probability that PPZ(ϕ) outputs assignment α depends on number of variables that are not forced. On the other hand variables that are forced correspond to at least one critical clause. Thus $\Pr[\text{PPZ}(\phi) = \alpha | \pi]$ improves if there are more critical clauses. With clever analysis it was shown in [10] that the success probability that one iteration of PPZ finds a satisfying assignment of ϕ is at least $2^{-n(1-1/k)}$ – which is at least $2^{-2n/3}$ for 3-CNF. Finally we note that PPZ makes one-sided error - if input formula ϕ is unsatisfiable then algorithm will always say so, but on satisfiable instances it may make error.

Let us consider another very simple randomized algorithm for 3-CNF. We will call this algorithm DEL (see Algorithm-2). In a single iteration of this algorithm we first delete one literal from each clause having three literals independently uniformly at random (a clause having less than three literals is ignored in this step) and obtain a new formula. Since input formula ϕ is a 3-CNF, we obtain a new formula ϕ' in 2-CNF for which there is a known linear time deterministic algorithm [3] (we will call this algorithm 2SAT). After running algorithm 2SAT(ϕ') if we find a satisfying assignment then we output that (after extending it to the rest of the variables (if any) - which can be assigned any truth value).

Again, let $\alpha \in S$ be a fixed solution in the set of all solutions of the input formula ϕ . Let $C(\alpha)$ be a critical clause of ϕ for variable x under solution α . Now observe that in the process of deletion if we delete the literal corresponding to variable x from $C(\alpha)$ then in the first step of the algorithm DEL(ϕ) we may produce a formula ϕ' having no satisfying

```

Algorithm DEL( $\phi$ ) Input: 3-CNF  $\phi = \bigwedge_{i=1}^m C_i$  on variables  $\{x_1, \dots, x_n\}$ 
for Each clause  $C$  having 3 literals do /* ignore clause with less than 3
literals */
| Select one literal uniformly at random and delete it.
end
Let  $\phi'$  be the obtained 2-CNF.
if 2SAT( $\phi'$ ) returns a satisfiable assignment  $\alpha$  then
| return  $\alpha$ .
else
| return "Unsatisfiable".
end

```

Algorithm 2: One iteration of procedure DEL(ϕ)

assignment (e.g. when α is the unique solution of formula ϕ , or if we make this error in a critical clause with respect to an isolated solution). Probability that this event does not happen is $2/3$ for $C(\alpha)$ - as a clause can not be critical for more than one variable, and every clause have 3 literals (other clauses with less than three literals were not considered in the deletion step). Now observe that only the deletion step of the algorithm DEL makes randomized choices, while executing the algorithm 2SAT on ϕ' is deterministic. Hence, if the deletion step of the algorithm makes no error (i.e. it does not remove solutions) then algorithm 2SAT on ϕ' will always find a satisfying assignment whenever input formula ϕ is satisfiable. Now assume there are $c(\alpha)$ number of critical clauses of ϕ under solution α . Then we have the probability that DEL(ϕ) returns a satisfying assignment with respect to an $\alpha \in S$ is $(2/3)^{c(\alpha)}$. In general $c(\alpha)$ can be polynomial in n , thus DEL performs well only when all satisfiable solutions of ϕ have less number of critical clauses. Let us note that like PPZ algorithm, DEL also makes one-sided error - if input formula ϕ is unsatisfiable then algorithm will always say so, but on satisfiable instances it may make error. This can be seen from the following: assume that the input formula ϕ is unsatisfiable but 2SAT(ϕ') returns with a satisfiable assignment - but ϕ' is obtained from ϕ by deleting one literal from each clause of size three, and hence the assignment that satisfies ϕ' also satisfies ϕ - a contradiction.

While success probability of DEL decreases with increasing number of critical clauses with respect to a fixed satisfiable solution α - success probability of PPZ increases. This fact suggests that a combination of these two algorithms can perform better. In order to motivate this further consider the worst case of PPZ algorithm [10] on 3-CNF. One such example is $\phi = \bigwedge_{i=0}^{m-1} (x_{3i+1} \oplus x_{3i+2} \oplus x_{3i+3})$ where $n = 3m$. Any solution α of ϕ has n critical clauses with respect to α , e.g. $\{(x_{3i+1} + \bar{x}_{3i+2} + \bar{x}_{3i+3}), (\bar{x}_{3i+1} + x_{3i+2} + \bar{x}_{3i+3}), (\bar{x}_{3i+1} + \bar{x}_{3i+2} + x_{3i+3})\}_{i=0}^{m-1}$, and success probability of PPZ on ϕ is $2^{-2n/3} \geq (1.5875)^{-n}$. On the other hand success probability of DEL on this instance is $(2/3)^n = (1.5)^{-n}$, and this is more than the success probability of PPZ. Our objective in this work is to combine these two algorithms such that the success probability of the combined algorithm is maximum of the success probability of DEL and PPZ on every input instance.

Organization. Rest of the paper is organized as follows. In section-2 we describe the algorithm DEL-PPZ - which is a combination of algorithm PPZ and algorithm DEL described before. Subsequently, in section-3 we analyze this combined algorithm. Finally in section-4 we conclude the paper.

2 Combined algorithm

In this section we describe the algorithm DEL-PPZ (see Algorithm-3) – which is a combination of the algorithm PPZ and algorithm DEL described above. Algorithm-3 describes one iteration, and in order to increase the success probability as a standard technique the algorithm needs to be executed several times. We will discuss about it at the end of this section. Like PPZ, one iteration of DEL-PPZ algorithm works by first selecting a random permutation of variables $\pi \in S_n$. Then for $i = 1, \dots, n$ the algorithm either execute steps that are similar to DEL(ϕ) and, if unsuccessful in finding a satisfying assignment, it execute steps that are similar to PPZ.

Algorithm DEL-PPZ(ϕ) Input: 3 – CNF $\phi = \bigwedge_{i=1}^m C_i$ on variables $\{x_1, \dots, x_n\}$
Pick a permutation π of the set $\{1, \dots, n\}$ uniformly at random.
 $\alpha := 0^n$
for $i = 1, \dots, n$ **do**
 for *Each clause C having 3 literals* **do** /* ignore clause with less than 3 literals */
 | Select one literal uniformly at random and delete it.
 end
 Let ϕ' be the obtained 2-CNF.
 if 2SAT(ϕ') *returns a satisfiable assignment* β **then**
 | (*) **return** β .
 else
 if *there is an unit clause corresponding to the variable $x_{\pi(i)}$* **then**
 | Set $x_{\pi(i)}$ so that corresponding unit clause is satisfied, let b be the assignment.
 else
 | Set $x_{\pi(i)}$ to TRUE or FALSE uniformly at random, let b be the assignment.
 end
 end
 $\phi := \phi[x_{\pi(i)} \leftarrow b]$, $\alpha_i := b$.
end
if α *is a satisfying assignment* **then**
 | (**) **return** α .
else
 | **return** “Unsatisfiable”.
end

Algorithm 3: One iteration of procedure DEL-PPZ(ϕ)

In other words, for each $i = 1, \dots, n$ the algorithm works on the current formula ϕ (like

PPZ, input formula ϕ is modified in every execution of the for loop as we assign truth value to variable $x_{\pi(i)}$ in i th execution) and first delete one literal from each clause of ϕ having three literals independently uniformly at random (a clause having less than three literals is ignored in this step) and obtain a new formula ϕ' . Since input formula ϕ is a 3-CNF, we obtain a new formula ϕ' in 2-CNF. After running algorithm 2SAT(ϕ') if we find a satisfying assignment then we output that (after extending it to the rest of the variables – which can be assigned any truth value), or else we again consider the current formula ϕ and assign truth values in $\{0, 1\}$ to variable $x_{\pi(i)}$. This is done as follows: we first check if there is an unsatisfied unit clause corresponding to variable $x_{\pi(i)}$ and force the value of $x_{\pi(i)}$ such that the corresponding unit clause gets satisfied, otherwise we assign truth values in $\{0, 1\}$ to $x_{\pi(i)}$ uniformly at random.

After this, the current formula ϕ is modified as $\phi := \phi[x_{\pi(i)} \leftarrow b]$. Where, by $\phi := \phi[x_{\pi(i)} \leftarrow b]$ we denote that variable $x_{\pi(i)}$ is assigned $b \in \{0, 1\}$, and formula ϕ is modified by treating each clause C of ϕ as follows: (i) if C is satisfied with this assignment then delete C , otherwise (ii) replace clause C by clause C' obtained by deleting any literals of C that are set to 0 by this assignment. Hence, DEL(ϕ) works on a new instance of formula in each execution of the for loop.

In every execution there are two places from where the algorithm could exit and return a satisfying assignment. When 2SAT(ϕ') returns a satisfying assignment β for some $i = 1, \dots, n$ (marked as (*), and we shall call it return by DEL) or at the end (marked as (**), which we shall call as return by PPZ).

It is not hard to see that the algorithm DEL-PPZ never returns an assignment if the input formula is unsatisfiable. As stated earlier, both PPZ and DEL has one-sided error and similar argument holds for DEL-PPZ as well. Thus the problem of interest would be to bound the probability that the algorithm answers “unsatisfiable” when the input formula ϕ is satisfiable. If $\tau(\phi)$ is the success probability of the algorithm DEL-PPZ on input ϕ , and if we execute the algorithm ω number of times, then for a satisfiable formula ϕ the error probability is equal to $(1 - \tau(\phi))^\omega \leq e^{-(\omega \cdot \tau(\phi))}$. This will be at most e^{-n} if we choose $\omega \geq n/\tau(\phi)$. In following section we shall estimate $\tau(\phi)$ and subsequently choose the value of ω .

3 Analysis of the combined algorithm

In this section we analyze the algorithm DEL-PPZ. Let $\phi = \bigwedge_{i=1}^m C_i$ be the input 3 – CNF formula defined on n variables $\{x_1, \dots, x_n\}$. Let $S \subseteq \{0, 1\}^n$ be the set of satisfying assignments of ϕ , $\alpha \in S$, and let π be any permutation in S_n .

Observe that in the main loop for each $i = 1, \dots, n$, the algorithm can return by DEL (marked as (*)) for any i . When the algorithm returns by DEL in the i th execution of the for loop, we estimate the success probability of obtaining any satisfying assignment in that execution of the for loop with respect to a $\alpha \in S$, for a fixed $\pi \in S_n$. Let us denote the i th such event by $A_i(\alpha)$ for $i = 1, \dots, n$ to indicate that i th execution returns by DEL with some satisfying assignment. To indicate that $\pi \in S_n$ is fixed we use the shorthand notation $\mathbf{Pr}[A|\pi]$ to denote $\mathbf{Pr}[A|\text{When } \pi \text{ is fixed}]$, for some event A . Also, for any event A let \bar{A} denote the complement of event A .

Similarly, let the event B denote that the algorithm returns by PPZ at the end of the for

loop (marked as (**)) and satisfying assignment returned is α , again for a fixed $\pi \in S_n$. Let us denote by $\text{DEL-PPZ}(\phi, \alpha)$ the event that with respect to some $\alpha \in S$, algorithm DEL-PPZ returns with a successful satisfying assignment - either by DEL or by PPZ. Now observe that the algorithm either returns by DEL in any one of the execution of the for loop for $i = 1 \dots, n$, or it returns by PPZ at the end of the for loop, hence, $\Pr[(\cup_{i=1}^n A_i(\alpha)) \cap B | \pi] = 0$. With this we have:

$$\begin{aligned} \Pr[\text{DEL-PPZ}(\phi, \alpha) | \pi] &= \Pr\left[\bigcup_{i=1}^n A_i(\alpha) \vee B | \pi\right] = \\ &= \left(\sum_{i=1}^n \Pr[A_i(\alpha) | \bigwedge_{j=1}^{i-1} \overline{A_j(\alpha)} \wedge \pi] \cdot \Pr\left[\bigwedge_{j=1}^{i-1} \overline{A_j(\alpha)} | \pi\right] \right) + \\ &+ \Pr[B | \bigwedge_{i=1}^n \overline{A_i(\alpha)} \wedge \pi] \cdot \Pr\left[\bigwedge_{i=1}^n \overline{A_i(\alpha)} | \pi\right] \end{aligned} \quad (1)$$

Recall, if the deletion step of the algorithm makes no error then algorithm 2SAT on ϕ' will always find a satisfying assignment. On the other hand in the process of deletion if we delete any unique true literal corresponding to a critical clause with respect to satisfying assignment α we may produce a formula ϕ' which will not have any satisfying assignment, and we will make error.

Let $c_\pi^{i-1}(\alpha)$ be the number of critical clauses of the resulting formula in the i th step with respect to assignment α on which the deletion step of DEL and subsequently 2SAT is executed. In specific $c_\pi^0(\alpha)$ denotes the number of critical clauses of the input formula ϕ . Since $c_\pi^{i-1}(\alpha)$ is the number of critical clauses of the resulting formula used in the i th step with respect to assignment α then success probability of returning by DEL in that step i.e. $\Pr[A_i(\alpha) | \bigwedge_{j=1}^{i-1} \overline{A_j(\alpha)} \wedge \pi]$ is $(2/3)^{c_\pi^{i-1}(\alpha)}$. Now for collection of events $\overline{A_1(\alpha)}, \dots, \overline{A_n(\alpha)}$ it holds that, for $r = 1, \dots, n$,

$$\begin{aligned} \Pr\left[\bigwedge_{j=1}^r \overline{A_j(\alpha)} | \pi\right] &= \Pr[\overline{A_1(\alpha)} | \pi] \cdot \Pr[\overline{A_2(\alpha)} | \overline{A_1(\alpha)} \wedge \pi] \cdot \dots \\ &\cdot \Pr[\overline{A_r(\alpha)} | \bigcap_{j=1}^{r-1} \overline{A_j(\alpha)} \wedge \pi] \end{aligned}$$

Observe that if the algorithm fails to return by DEL in the $(r-1)$ th execution of the for loop, then given there were $c_\pi^{r-2}(\alpha)$ many critical clauses in the beginning of the $(r-1)$ th execution, there will be $c_\pi^{r-1}(\alpha)$ many critical clauses after PPZ part of the algorithm executes. Hence, for $r = 1, \dots, n$, given all $(r-1)$ trial of return by DEL has failed we have:

$$\Pr[\overline{A_r(\alpha)} | \bigcap_{j=1}^{r-1} \overline{A_j(\alpha)} \wedge \pi] = \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{r-1}(\alpha)}\right), \text{ for } r = 1, \dots, n.$$

Hence,

$$\Pr\left[\bigwedge_{j=1}^r \overline{A_j(\alpha)} | \pi\right] = \prod_{j=1}^r \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{j-1}(\alpha)}\right), \text{ for } r = 1, \dots, n.$$

And we have,

$$\begin{aligned} \sum_{i=1}^n \Pr[A_i(\alpha) | \bigwedge_{j=1}^{i-1} \overline{A_j(\alpha)} \wedge \pi] \cdot \Pr[\bigwedge_{j=1}^{i-1} \overline{A_j(\alpha)} | \pi] = \\ \sum_{i=1}^n \left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)} \cdot \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{j-1}(\alpha)}\right) \end{aligned} \quad (2)$$

Let $d_\pi(\alpha)$ be the number of variables that are *not* forced by PPZ. Then we have:

$$\Pr[B | \bigwedge_{i=1}^n \overline{A_i(\alpha)} \wedge \pi] \cdot \Pr[\bigwedge_{i=1}^n \overline{A_i(\alpha)} | \pi] = 2^{-d_\pi(\alpha)} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)}\right) \quad (3)$$

Using Eq. (2) and Eq. (3) with Eq. (1) it is easy to see now that,

$$\begin{aligned} \Pr[\text{DEL-PPZ}(\phi, \alpha) | \pi] = \sum_{i=1}^n \left(\left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)} \cdot \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{j-1}(\alpha)}\right) \right) + \\ \left(2^{-d_\pi(\alpha)} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)}\right) \right) \end{aligned} \quad (4)$$

Let $\mathbf{Exp}_\pi[X]$ denote the expectation of random variable X taken over all random permutation $\pi \in S_n$. Now it is easy to see that using Eq. (4), and summing over the set S of all satisfying solutions of ϕ , we have using linearity of expectation:

$$\begin{aligned} \tau(\phi) &= \Pr[\text{DEL-PPZ}(\phi) \text{ outputs some satisfying assignment}] \\ &= \sum_{\alpha \in S} \mathbf{Exp}_\pi \left[\sum_{i=1}^n \left(\left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)} \cdot \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{j-1}(\alpha)}\right) \right) \right] + \\ &\quad \sum_{\alpha \in S} \mathbf{Exp}_\pi \left[2^{-d_\pi(\alpha)} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3}\right)^{c_\pi^{i-1}(\alpha)}\right) \right] \\ &\geq \sum_{\alpha \in S} \left[\sum_{i=1}^n \left(\left(\frac{2}{3}\right)^{\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)]} \cdot \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3}\right)^{\mathbf{Exp}_\pi[c_\pi^{j-1}(\alpha)]}\right) \right) \right] + \\ &\quad \sum_{\alpha \in S} \left[2^{-\mathbf{Exp}_\pi[d_\pi(\alpha)]} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3}\right)^{\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)]}\right) \right] \end{aligned} \quad (5)$$

Where last inequality (Eq. (5)) follows from Jensen's inequality (cf. [19]) - which states that for a random variable $X = (c_\pi^0(\alpha), c_\pi^1(\alpha), \dots, c_\pi^{n-1}(\alpha), d_\pi(\alpha))$ and any convex function f , $\mathbf{Exp}[f(X)] \geq f(\mathbf{Exp}[X])$. Now observe that $c_\pi^0(\alpha), c_\pi^1(\alpha), \dots, c_\pi^{n-1}(\alpha)$ is a non-increasing sequence of integers, i.e. $c_\pi^0(\alpha) \geq c_\pi^1(\alpha) \geq \dots \geq c_\pi^{n-1}(\alpha)$, because in every execution whenever a variable is forced by PPZ a collection of critical clause gets satisfied and are removed from ϕ . Hence, we can simplify Eq. (5) as follows using the fact that $\mathbf{Exp}_\pi[c_\pi^0(\alpha)] =$

$c_\pi^0(\alpha)$, when $c_\pi^0(\alpha) \neq 0$. On the other hand when $c_\pi^0(\alpha) = 0$, it follows that $\tau(\phi) = 1$ by taking $c_\pi^{i-1}(\alpha) = 0$ for all $i = 1, \dots, n$ in Eq. (4):

$$\begin{aligned} \tau(\phi) \geq \sum_{\alpha \in S} & \left[\left(\frac{2}{3} \right)^{c_\pi^0(\alpha)} \cdot \sum_{i=1}^n \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3} \right)^{\mathbf{Exp}_\pi[c_\pi^{j-1}(\alpha)]} \right) \right] + \\ & \sum_{\alpha \in S} \left[2^{-\mathbf{Exp}_\pi[d_\pi(\alpha)]} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3} \right)^{\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)]} \right) \right] \end{aligned} \quad (6)$$

Let $l(\alpha) \triangleq |\{\alpha' \in S : d(\alpha, \alpha') = 1\}|$ denote that number of satisfying assignments of ϕ that has Hamming distance 1 from α . Using arguments from [10] (cf. [20]) we can bound $\mathbf{Exp}_\pi[d_\pi(\alpha)]$. For completeness we state it here. Given the definition of $l(\alpha)$, there are $n - l(\alpha)$ variables such that each of them appear as a unique true literal in some critical clause of ϕ . It follows that each such variable $x_{\pi(i)}$ will be forced under randomly chosen $\pi \in S_n$ if $x_{\pi(i)}$ occurs last in the corresponding critical clause. This happens with probability at least $1/3$. Using linearity of expectation we have that expected number of forced variables is at least $((n - l(\alpha)))/3$, and hence,

$$\mathbf{Exp}_\pi[d_\pi(\alpha)] \leq \left(n - \frac{(n - l(\alpha))}{3} \right) \quad (7)$$

Now we concentrate on giving bounds on $\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)]$ for $i = 1, \dots, n$. Let $\mathcal{C}(\alpha)$ be the set of all critical clauses of ϕ with respect to α . Let us also denote by $r_\pi^i(\alpha)$ the number of critical clauses that are removed by PPZ at the end of i th execution of the for loop. Clearly the expected number of critical clauses in the beginning of the i th execution of the for loop, $\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)]$ is equal to the expected number of critical clauses that were present in the beginning of the $(i-1)$ th execution minus the expected number of critical clauses that were removed by PPZ at the end of the $(i-1)$ th execution. It follows, $\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)] = \mathbf{Exp}_\pi[c_\pi^{i-2}(\alpha)] - \mathbf{Exp}_\pi[r_\pi^{i-1}(\alpha)]$, with $\mathbf{Exp}_\pi[c_\pi^0(\alpha)] = c_\pi^0(\alpha)$. Let $\mathcal{C}_\pi^{i-2}(\alpha)$ denote the set of all critical clauses in the beginning of $i-1$ th execution of the for loop. Also, let X_c be an indicator random variable taking values in $\{0, 1\}$ such that $X_c = 1$ iff clause $c \in \mathcal{C}_\pi^{i-2}(\alpha)$ is removed by the end of $i-1$ th execution of the for loop. Using linearity of expectation we have that,

$$\begin{aligned} \mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)] &= c_\pi^{i-2}(\alpha) - \sum_{c \in \mathcal{C}_\pi^{i-2}(\alpha)} \mathbf{Exp}_\pi[X_c] \\ &= c_\pi^{i-2}(\alpha) - \sum_{c \in \mathcal{C}_\pi^{i-2}(\alpha)} (1 \cdot \mathbf{Pr}_\pi[X_c = 1] + 0 \cdot \mathbf{Pr}_\pi[X_c = 0]) \\ &= c_\pi^{i-2}(\alpha) - \sum_{c \in \mathcal{C}_\pi^{i-2}(\alpha)} \mathbf{Pr}_\pi[X_c = 1] \end{aligned}$$

As discussed above, a clause can not be critical for more than one variable. On the other hand each variable $x_{\pi(i)}$ that appears as a unique true literal in some set of critical clauses of ϕ creates a partition of $\mathcal{C}(\alpha)$. Let us denote the cardinality of the partition of critical clauses corresponding to variable $x_{\pi(i)}$ with respect to α by $t_\pi^i(\alpha)$ (where, $t_\pi^0(\alpha) = 0$).

Surely, $c_\pi^0(\alpha) = \sum_{i=1}^n t_\pi^i(\alpha)$. Now in the $(i-1)$ th execution we consider variable $x_{\pi(i-1)}$, that appears as a unique true literal in $t_\pi^{i-1}(\alpha)$ many critical clauses under assignment α . There is one possible way a critical clause c is removed by PPZ in accordance with assignment α under randomly chosen $\pi \in S_n$ - (as discussed above) when corresponding variable is forced, and probability of that event to occur for clause c is at least $1/3$.

Note that here we have ignored one particular effect of the statement $\phi := \phi[x_{\pi(i)} \leftarrow b]$. By this modification of ϕ in every execution of the for loop a critical clause with 3 literals can become a clause having 2 or less number of literals and still remain critical - but will not be considered in the deletion step in next execution of the for loop. However, considering this effect will only improve the success probability of return by DEL, as there will be lesser number of critical clauses in the subsequent execution of the for loop, on the other hand it will make the analysis complicated.

Based on the above discussion we have, $\sum_{c \in \mathcal{C}_\pi^{i-2}(\alpha)} \Pr_\pi[X_c = 1] \geq t_\pi^{i-1}(\alpha)/3$. And we have, $\mathbf{Exp}_\pi[c_\pi^0(\alpha)] = c_\pi^0(\alpha)$, and $\mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)] \leq c_\pi^{i-2}(\alpha) - \frac{1}{3} \cdot t_\pi^{i-1}(\alpha)$. Solving this we obtain that,

$$\begin{aligned} \mathbf{Exp}_\pi[c_\pi^{i-1}(\alpha)] &\leq c_\pi^0(\alpha) - \frac{1}{3} \cdot \sum_{j=1}^{i-1} t_\pi^j(\alpha) = c_\pi^0(\alpha) - \frac{1}{3} \cdot \sum_{j=1}^n t_\pi^j(\alpha) + \frac{1}{3} \cdot \sum_{j=i}^n t_\pi^j(\alpha) \\ &= c_\pi^0(\alpha) - \frac{c_\pi^0(\alpha)}{3} + \frac{1}{3} \cdot \sum_{j=i}^n t_\pi^j(\alpha) = \frac{2}{3} \cdot c_\pi^0(\alpha) + \frac{1}{3} \cdot \sum_{j=i}^n t_\pi^j(\alpha) \end{aligned} \quad (8)$$

In following, we simplify notation by replacing $c_\pi^0(\alpha)$ with $c(\alpha)$, and $t_\pi^i(\alpha)$ with $t^i(\alpha)$. Now observe that in the expression $\prod_{j=1}^{i-1} (1 - (2/3)^{\mathbf{Exp}_\pi[c_\pi^{j-1}(\alpha)]})$ in Eq. (6), for every i , term $(1 - (2/3)^{\mathbf{Exp}_\pi[c_\pi^0(\alpha)]})$ appears in every product. Also observe that $\sum_{j=i}^n t_\pi^j(\alpha) \geq 0$ for any i . So we use $\prod_{j=1}^{i-1} (1 - (2/3)^{\frac{2}{3} \cdot c(\alpha)}) \leq \prod_{j=1}^{i-1} (1 - (2/3)^{\mathbf{Exp}_\pi[c_\pi^{j-1}(\alpha)]})$ as lower bound, and with Eq. (7) and Eq. (8) we modify Eq. (6) as follows:

$$\begin{aligned} \tau(\phi) &\geq \sum_{\alpha \in S} \left[\left(\frac{2}{3} \right)^{c(\alpha)} \cdot \sum_{i=1}^n \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3} \right)^{\frac{2}{3} \cdot c(\alpha)} \right) \right] + \\ &\quad \sum_{\alpha \in S} \left[2^{-\left(n - \frac{(n-l(\alpha))}{3} \right)} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3} \right)^{\frac{2}{3} \cdot c(\alpha)} \right) \right] \end{aligned} \quad (9)$$

Now observe that since α is $(n-l(\alpha))$ -isolated it must be that $c(\alpha) \geq (n-l(\alpha))$. In fact recall that $c(\alpha) = \sum_{i=1}^n t^i(\alpha)$. Let us define t as the minimum of $t^i(\alpha)$ over $i \in \{1, \dots, n\}$ such that $x_{\pi(i)}$ appears as unique true literal in at least one critical clause, and $T_{av} \triangleq c(\alpha)/(n-l(\alpha))$. We have $T_{av}(n-l(\alpha)) = c(\alpha) \geq t(n-l(\alpha))$. Also note that $t \geq 1$. Using these two facts

with Eq. (9), we can lower bound $\tau(\phi)$ now as follows¹:

$$\begin{aligned}
\tau(\phi) &\geq \sum_{\alpha \in S} \left[\left(\frac{2}{3} \right)^{T_{av}(n-l(\alpha))} \cdot \sum_{i=1}^n \prod_{j=1}^{i-1} \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right) \right] + \\
&\quad \sum_{\alpha \in S} \left[2^{-\left(n - \frac{(n-l(\alpha))}{3} \right)} \cdot \prod_{i=1}^n \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right) \right] \\
&= 2^{-T_{av} \cdot n \cdot \log(3/2)} \cdot \sum_{\alpha \in S} \left[\left(\frac{2}{3} \right)^{-T_{av} \cdot l(\alpha)} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right)^{n-1} \right] + \\
&\quad 2^{-\frac{2n}{3}} \cdot \sum_{\alpha \in S} \left[2^{-\frac{l(\alpha)}{3}} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right)^n \right] \tag{10}
\end{aligned}$$

Let $L \triangleq \mathbf{Exp}_{\alpha \in S}[l(\alpha)]$ and $s \triangleq |S|$. Using Jensen's inequality we obtain:

$$\begin{aligned}
&\sum_{\alpha \in S} \left[\left(\frac{2}{3} \right)^{-T_{av} \cdot l(\alpha)} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right)^{n-1} \right] \geq \\
&\quad s \cdot \left(\frac{2}{3} \right)^{-T_{av} \cdot L} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-L)}{3}} \right)^{n-1} \tag{11}
\end{aligned}$$

And,

$$\sum_{\alpha \in S} \left[2^{-\frac{l(\alpha)}{3}} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-l(\alpha))}{3}} \right)^n \right] \geq s \cdot 2^{-L/3} \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-L)}{3}} \right)^n \tag{12}$$

Combining Eq. (11) and Eq. (12) with Eq. (10) we have:

$$\begin{aligned}
\tau(\phi) &\geq s \cdot \left(\frac{2^{-(n-L) \cdot T_{av} \cdot \log(3/2)}}{\left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-L)}{3}} \right)} + 2^{-(2n+L)/3} \right) \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-L)}{3}} \right)^n \\
&\geq s \cdot \left(2^{-(n-L) \cdot T_{av} \cdot \log(3/2) + o(1)} + 2^{-(2n+L)/3} \right) \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n-L)}{3}} \right)^n \tag{13}
\end{aligned}$$

In order to bound L we will use the edge isoperimetric inequality from [21], which states that for any $S \subseteq \{0, 1\}^n$, $|\{(a, a') | a, a' \in S \text{ and } d(a, a') = 1\}| \leq |S| \cdot \log(|S|)$, and $\sum_{\alpha \in S} l(\alpha) \leq s \cdot \log s$. So using this result as in [20] $L = \mathbf{Exp}_{\alpha \in S}[l(\alpha)] \leq \log s$. On the other hand, it is not hard to observe that the lower bound on $\sum_{\alpha \in S} l(\alpha)$ is 0 as long as $s \leq 2^{n-1}$. This

¹All logarithms are base 2.

can be seen as follows. We consider $\{0, 1\}^n$ as the vertex set of a graph (Hamming cube, denoted Q_n) and for $a, a' \in \{0, 1\}^n$, aa' is an edge of this graph iff $d(a, a') = 1$. Now the lower bound in question corresponds to finding a subgraph of Q_n having s many vertices and having minimum number of induced edges having both of their end-points in $S \subseteq \{0, 1\}^n$. Now observe that since Q_n is bipartite, with $s \leq 2^{n-1}$ we have always a set of vertices of size s having no edges between them. Updating Eq. (13) with this we have:

$$\begin{aligned} \tau(\phi) &\geq s \cdot \left(2^{-n \cdot T_{av} \cdot \log(3/2)} + 2^{-(2n + \log s)/3} \right) \cdot \left(1 - \left(\frac{2}{3} \right)^{\frac{2t(n - \log s)}{3}} \right)^n \\ &= \left(s \cdot 2^{-n \cdot T_{av} \cdot \log(3/2)} + (2^{-n} \cdot s)^{2/3} \right) \cdot \left(1 - (2^{-n} \cdot s)^{2/3 \cdot t \cdot \log(3/2)} \right)^n \end{aligned} \quad (14)$$

Now it can be seen that the term, $(1 - (2^{-n} \cdot s)^{2/3 \cdot t \cdot \log(3/2)})^n$ converges to 1 very fast with n . So for sufficiently large n we can ignore this term. Thus for sufficiently large n we have from Eq. (14),

$$\tau(\phi) \geq \left(s \cdot 2^{-n \cdot T_{av} \cdot \log(3/2)} + (2^{-n} \cdot s)^{2/3} \right) \quad (15)$$

Lower bound on $\tau(\phi)$ from Eq. (15) shows that (like PPZ [10]) performance of the algorithm DEL-PPZ improves with more number of solutions. On the other hand for any value of $1 \leq T_{av} < 2/(3 \cdot \log(3/2)) = 1.13967$, performance of DEL-PPZ is better than PPZ. For higher values of T_{av} and with $s = 1$ performance of the algorithm DEL-PPZ tends to become same as the performance of PPZ algorithm, which is 1.5875^{-n} . On the other hand for $s = 1$ (unique solution) and $T_{av} = 1$ (one critical clause per variable) performance of the algorithm DEL-PPZ tends to become same as the performance of algorithm DEL, which is 1.5^{-n} (see Fig. 1.). Our results on the algorithm DEL-PPZ can now be summarized in the following

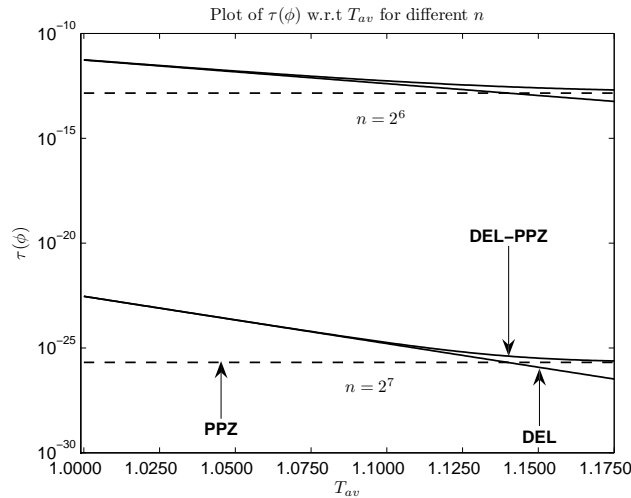


Figure 1: Illustration of how success probability of PPZ, DEL and DEL-PPZ changes with $1 \leq T_{av} < 2/(3 \cdot \log(3/2))$ for different values of n with $s = 1$ (Y-axis is in log scale).

statements:

Lemma 3.1. *Let ϕ be any 3-CNF formula over n variables that has s number of satisfying assignments, and let T_{av} be the average number of clauses per variable that appear as unique true literal in one or more critical clauses in ϕ . Then probability that one iteration of algorithm DEL-PPZ outputs some satisfying assignment is at least,*

$$\left(s \cdot 2^{-n \cdot T_{av} \cdot \log(3/2)} + (2^{-n} \cdot s)^{2/3} \right)$$

Theorem 3.1. *Let ϕ be any 3-CNF formula over n variables and let $T_{av} \in [1, 2/(3 \cdot \log(3/2))]$ be the average number of clauses per variable that appear as unique true literal in one or more critical clauses in ϕ . Then probability that one iteration algorithm DEL-PPZ outputs some satisfying assignment is at least 1.5^{-n} for $T_{av} = 1$ and decreases to 1.5875^{-n} for $T_{av} = 2/(3 \cdot \log(3/2))$. For $T_{av} > 2/(3 \cdot \log(3/2))$ probability that one iteration algorithm DEL-PPZ outputs some satisfying assignment is at least 1.5875^{-n} . And these bounds are tight for $\phi = \bigwedge_{i=1}^{m-1} (x_{3i} \oplus x_{3i+1} \oplus x_{3i+2})$ where $n = 3m$.*

Now recall that we can also bound the error probability of the algorithm to $o(1)$ if we execute the algorithm DEL-PPZ for $\omega \geq n/\tau(\phi)$ times. With this we obtain following results:

Theorem 3.2. *Let $T_{av} \geq 1$ be a real number. There is a randomized algorithm for 3-SAT, namely DEL-PPZ, that given any 3-CNF formula ϕ over n variables with s number of satisfying assignments, makes one sided error of at most $o(1)$ on satisfiable instances, otherwise outputs one of the satisfying assignments of ϕ in expected time*

$$\mathcal{O} \left(\min \left\{ \left(\text{poly}(n) \cdot \left(\frac{2^{n \cdot T_{av} \cdot \log(3/2)}}{s} \right) \right), \left(\text{poly}(n) \cdot \left(\frac{2^n}{s} \right)^{2/3} \right) \right\} \right)$$

4 Concluding remarks

As stated in the introduction that recently best known randomized bound for 3-SAT is $\mathcal{O}(1.32216^n)$ [16]. It is interesting to note that this algorithm is a combination of the random walk algorithm of [13, 6] and algorithm of [11] (we will call this algorithm PPSZ), and success probability of algorithm in [16] is maximum of the success probability of random walk algorithm of [13, 6] and algorithm PPSZ. Algorithm PPSZ is a combination of 3^d bounded resolution on input 3-CNF formula ϕ followed by the PPZ algorithm. Purpose of using a bounded resolution first is to increase the success probability of PPZ algorithm - by increasing the number of critical clauses per variable - as that will in effect increase the probability that a variable (that appears as unique true literal in a set of critical clauses) is forced with respect to a randomly chosen permutation. On the other hand algorithm DEL-PPZ performs better when the average number of critical clause per variable in ϕ is close to 1. We believe that for values of T_{av} close to 1 our algorithm improves the algorithm PPSZ and best known randomized bound for 3-SAT as presented in [16]. We will consider this analysis as our future work.

References

- [1] Cook, S.A.: The complexity of theorem-proving procedures. In: STOC '71: Proceedings of the third annual ACM symposium on Theory of computing, New York, NY, USA, ACM Press (1971) 151–158
- [2] Levin, L.: Universal'nyie perebornyie zadachi (universal search problems: in russian). Problemy Peredachi Informatsii, English translation in [22] **9**(3) (1973) 265–266
- [3] Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Process. Lett. **8**(3) (1979) 121–123
- [4] Dantsin, E.: Two propositional proof systems based on the splitting method. Zapiski Nauchnykh Seminarov LOMI, 105:24-44, 1981. (in Russian), English translation in Journal of Soviet Mathematics **22**(3) (1983) 1293–1305
- [5] Monien, B., Speckenmeyer, E.: Solving satisfiability in less than 2^n steps. Discrete Applied Mathematics **10** (1985) 287–295
- [6] Schöning, U.: A probabilistic algorithm for k -SAT based on limited local search and restart. Algorithmica **32**(4) (2002) 615–623
- [7] Dantsin, E., Goerdt, A., Hirsch, E.A., Kannan, R., Kleinberg, J., Papadimitriou, C., Raghavan, P., Schöning, U.: A deterministic $(2 - 2/(k + 1))^n$ algorithm for k -SAT based on local search. Theor. Comput. Sci. **289**(1) (2002) 69–83
- [8] Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k -SAT. J. ACM **52**(3) (2005) 337–364
- [9] Kullmann, O.: New methods for 3-SAT decision and worst-case analysis. Theor. Comput. Sci. **223**(1-2) (1999) 1–72
- [10] Paturi, R., Pudlák, P., Zane, F.: Satisfiability coding lemma. Chicago Journal of Theoretical Computer Science **1999**(115) (1999)
- [11] Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k -SAT. In: FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (1998) 628
- [12] Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM **5**(7) (1962) 394–397
- [13] Schöning, U.: A probabilistic algorithm for k -SAT and constraint satisfaction problems. In: FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (1999) 410
- [14] Hofmeister, T., Schöning, U., Schuler, R., Watanabe, O.: Randomized algorithms for 3-SAT. Theor. Comp. Sys. **40**(3) (2007) 249–262

- [15] Iwama, K., Tamaki, S.: Improved upper bounds for 3-SAT. In: SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2004) 328–328
- [16] Rolf, D.: Improved bound for the PPSZ/Schöning-algorithm for 3-SAT. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* **1**(1) (2006) 111–122
- [17] Brueggemann, T., Kern, W.: An improved deterministic local search algorithm for 3-SAT. *Theor. Comput. Sci.* **329**(1-3) (2004) 303–313
- [18] Rolf, D.: Derandomization of PPSZ for unique k -SAT. In Bacchus, F., Walsh, T., eds.: SAT. Volume 3569 of *Lecture Notes in Computer Science.*, Springer (2005) 216–225
- [19] Feller, W.: *An Introduction to Probability Theory and Its Applications*. Second edn. Volume II. John Wiley and Sons, New York (1971)
- [20] Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique k -SAT: An isolation lemma for k -CNFs. *J. Comput. Syst. Sci.* **74**(3) (2008) 386–393
- [21] Harper, L.H.: A necessary condition on minimal cube numberings. *Journal of Applied Probability* **4**(2) (1967) 397–401
- [22] Trakhtenbrot, B.A.: A survey of russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing* **6**(4) (1984) 384–400